Subject: IDL enhancements (was Re: idl2matlab translate-o-matic)
Posted by Martin Schultz on Fri, 25 Feb 2000 08:00:00 GMT
View Forum Message <> Reply to Message

... while we are "working" at getting IDL to be more c-like, I'd give a lot to save
my fingers from RSI (repetative stress injury for the unitiated) -- although I'd
probably use the saved keywtrokes for newsgroup messages anyhow ;-)

Here two or three things which I would favor very much:

1.) shorten this extra verbose if then endif else endelse structure:
```
  if CONDITION
  else
  endif (why not even 'fi')
```

  is completely sufficient to make code legible. It shouldn't be too hard to
  allow backward compability here, because they would just have to ignore the
  useless extra tokens. One line statements would be:

```
  if CONDITION  STATEMENT
```

  If no statement follows on the same line, it must be a block-if construct and
  needs to be terminated by endif. Isn't too complicated, is it?

2.) allow assignment statements inside if's or while's:

```
  a = 100
  b =   5
  while (i=a-b gt 50) do a=a-b
```

  (This actually compiles but doesn't produce what you would like it to.)

  OR:
```
  if (p=strpos(s='The quick brown fox ...','fox') ge 0) then begin
    half1 = strmid(s,0,p)
    half2 = strmid(s,p+1,*)  ; the '*' would be another nice feature !
    ...
  endif
```

  I guess one has to be careful with more complicated conditional clauses such as

if ((i=a-b gt 50) OR (j=a+b lt 200)) then ...
for then j could be undefined when you need it. But shouldn't that be
a
programmer's responsibility? After all: one doesn't have to use
features like this
if one doesn't want to.

On the other side: Congratulations to RSI for introducing regular
expression searching!
Now, perhaps, I won't have to learn too much pearl after all ;-)
Sad thing though: it'll probably take another year at least before a
majority of
users has upgraded to 5.3...

Cheers,
Martin

"J.D. Smith" wrote:
>
> Craig Markwardt wrote:
>>
>> "J.D. Smith" <jdsmith@astro.cornell.edu> writes:
>>>> Since I can't pass a testing function to that routine (IDL doesn't have
>>>> higher order functions), I will accept a routine, for illustrative purposes,
>>>> that removes all even values from the array.
>>>>
>>>> Now suppose some joker passes an array containing only even values to that
>>>> routine...
>>>>
>>>> - DM
>>>>
>>>
>>> wh=where(array mod 2, cnt)
>>> if cnt gt 0 then return,array else return, -1
>>>
>>> I use scalars (often -1) as cheap and easy to use empty arrays. Anything with:
>>>
>>> size(x,/N_DIMEN) eq 0
>>>
>>> is patently *not* an array.
>>>
>>>
>>> And as far as the lack of "higher order testing functions":
>>>
>>> function evens, arr

>>>   return, arr mod 2 eq 0
>>> end
>>>
>>> function odds, arr
>>>   return, arr mod 2
>>> end
>>>
>>> function exclude,arr, exc_func
>>>   wh=where(call_function(exc_func,arr) eq 0,cnt)
>>>   if cnt gt 0 then return,arr else return,-1
>>> end
>>>
>>> and to get rid of the odds, e.g.:
>>>
>>> IDL> a=exclude(b,"odds")
>>
>> Okay, but let's say now you wanted to merge two lists like that
>> together. Wouldn't this be nice:
>>
>> IDL> c =
>>
>> The way I say it makes it sound like it's just an inconvenience, which
>> it is. But for gosh sakes, its a *completeness* issue too. We don't
>> have a general purpose number system without zero! It would be silly.
>> Why should we have lists without the empty list? Instead we have to
>> drag around this extra notion of the COUNT or play tricks by returning
>> scalars.
>>
>
> I totally agree with you about the convenience of such an entity. I'm not
> trying to deny that. What I'm trying to show is that what we might think of as
> an "empty array" or "empty list" is just an abstract notion, actually
> implemented in code in some way analogous to what I've done. In stark contrast
> to the issue you raise of a complete number system, in which the internal
> representation for "0" is equivalent to that for any other number, an empty
> array is achieved only by special case programming, which just happens to be
> hidden from our sight. Now, IDL is not C, and lots of special case programming
> is hidden from our sight, so I'm certainly not arguing that hidden conveniences,
> if well implemented, are to be avoided. I just want everyone to understand that
> this would be an addition purely motivated by convenience, and that there really
> is no fundamental "incompleteness".
>
> Having said that, I see no reason that it couldn't be done pretty easily.
> Variables can already be marked "undefined", so why not extend that somewhat and
> allow "undefined" arrays and lists to exist. Dimensionality is important of
> course, so the concept of a 2x2 empty array need be addressed, etc., but I
> wouldn't think it's prohibitive.
>

> JD
>
>
> --
> J.D. Smith |*| WORK: (607) 255-5842
> Cornell University Dept. of Astronomy |*| (607) 255-6263
> 304 Space Sciences Bldg. |*| FAX: (607) 255-5875
> Ithaca, NY 14853 |*|

--
[[[[[[[[[
[[Dr. Martin Schultz Max-Planck-Institut fuer Meteorologie [[
[[Bundesstr. 55, 20146 Hamburg [[
[[phone: +49 40 41173-308 [[
[[fax: +49 40 41173-298 [[
[[martin.schultz@dkrz.de [[
[[[[[[[[[